

Microsoft DevOps Solutions: Designing and Implementing Health Checks

CLOUD APPLICATION HEALTH



David Tucker

TECHNICAL ARCHITECT & CTO CONSULTANT

@_davidtucker_ davidtucker.net

Objectives

Cloud Application Health

- Analyze system dependencies to determine which dependency should be included in health check
- Calculate healthy response timeouts based on SLO for the service
- Design approach for partial health situations
- Design approach for piecemeal recovery (e.g. to improve recovery time objective strategies)

Objectives

Monitoring Compute Health

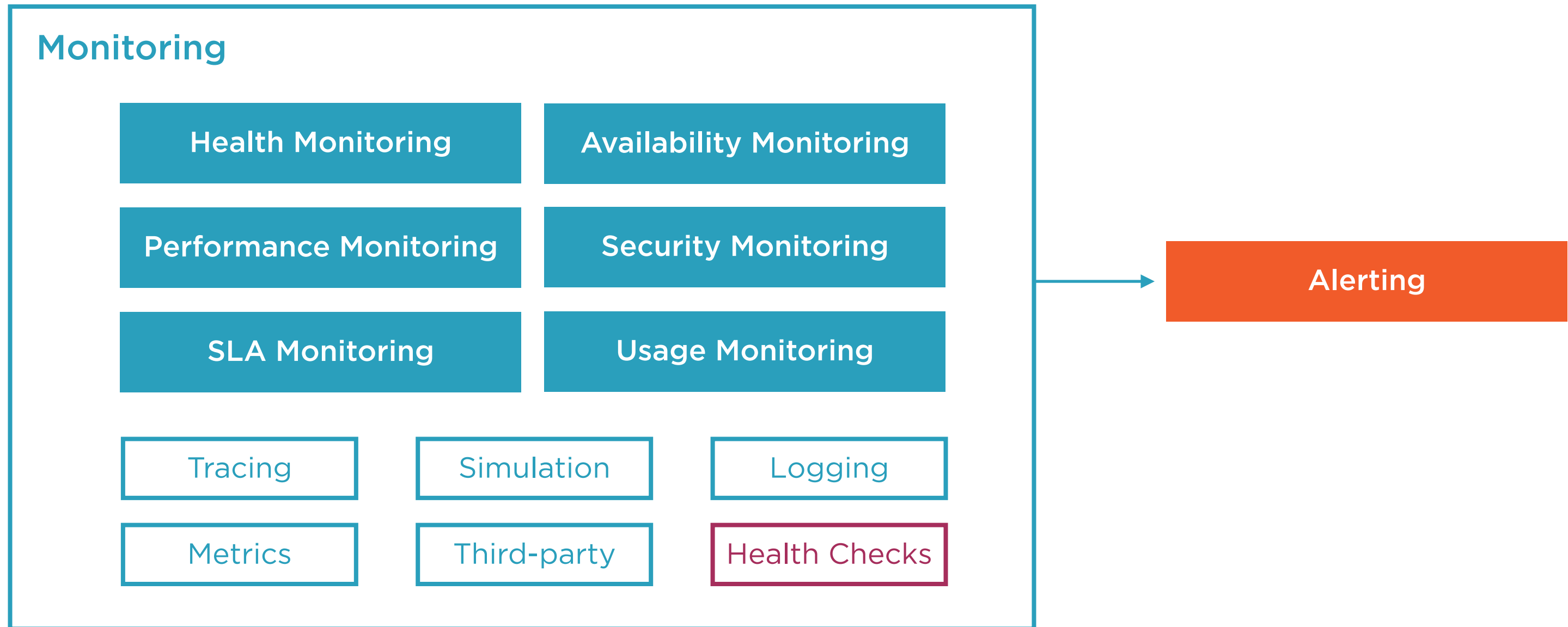
- Integrate health check with compute environment
- Implement different types of health checks (liveness, startup, shutdown)

Role of Monitoring and Health Checks

“A system is **healthy** if it is running and capable of processing requests.”

Microsoft Azure Documentation

Supporting Cloud Applications

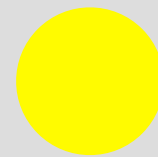


Application Health States



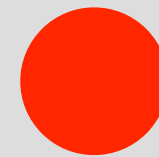
Green

All monitored
systems are working
as intended



Yellow

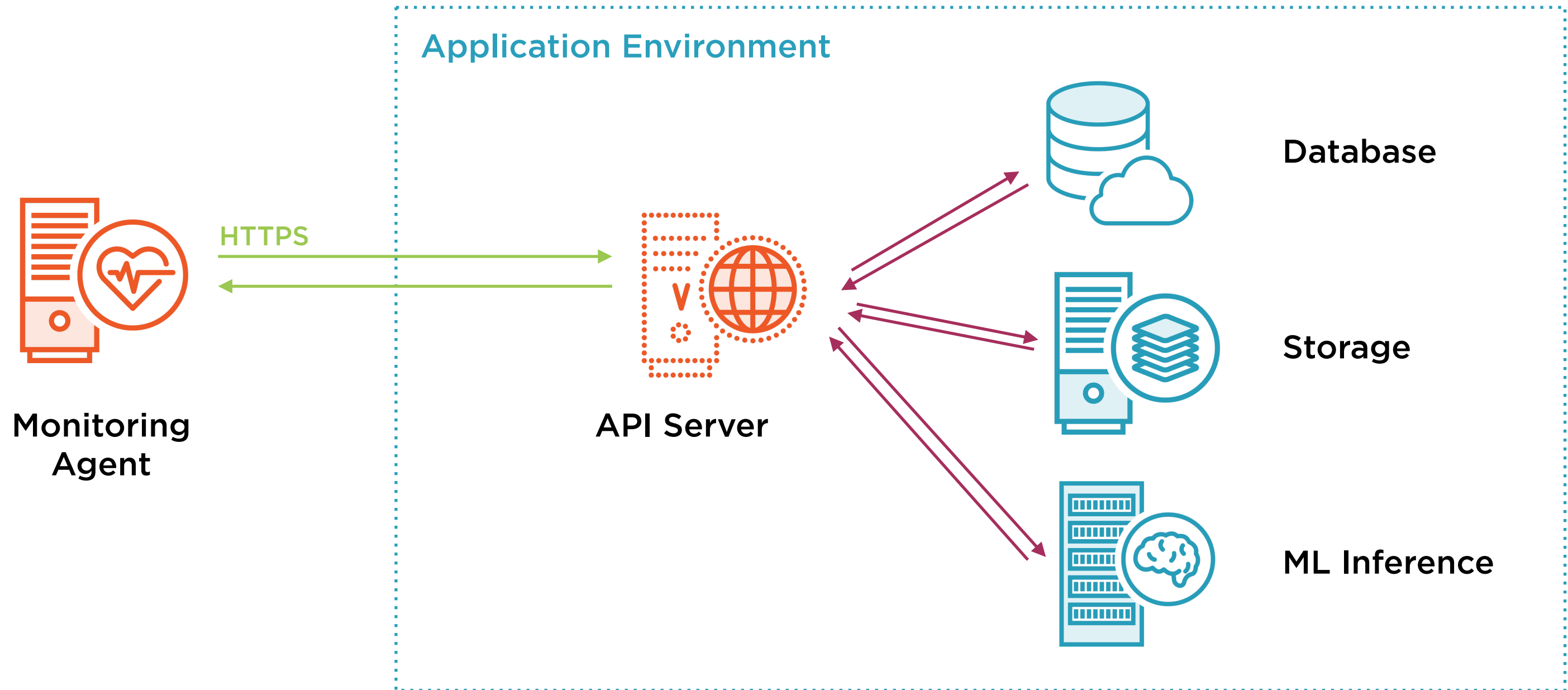
Partially healthy with
some affected
systems



Red

Application is
non-functional or
stopped

Anatomy of a Health Check



Health Check Elements

Health Check Endpoint

The checks performed by your application through a health check endpoint

Result Interpreter

The system calling and interpreting the health check response

Health Check Response

Response Time: 114 ms

200 OK

```
{  
  totalDuration: 86,  
  services: {  
    storage: 45,  
    database: 25,  
    inference: 42  
  },  
  status: "green"  
}
```

Application Health Points

Health Check Response



Storage Service



Database Service



Machine Learning Inference Service



“A **service-level agreement** (SLA) is a commitment between a service provider and a client. Service-level agreements can contain numerous **service-performance metrics** with corresponding **service-level objectives**.”

Wikipedia

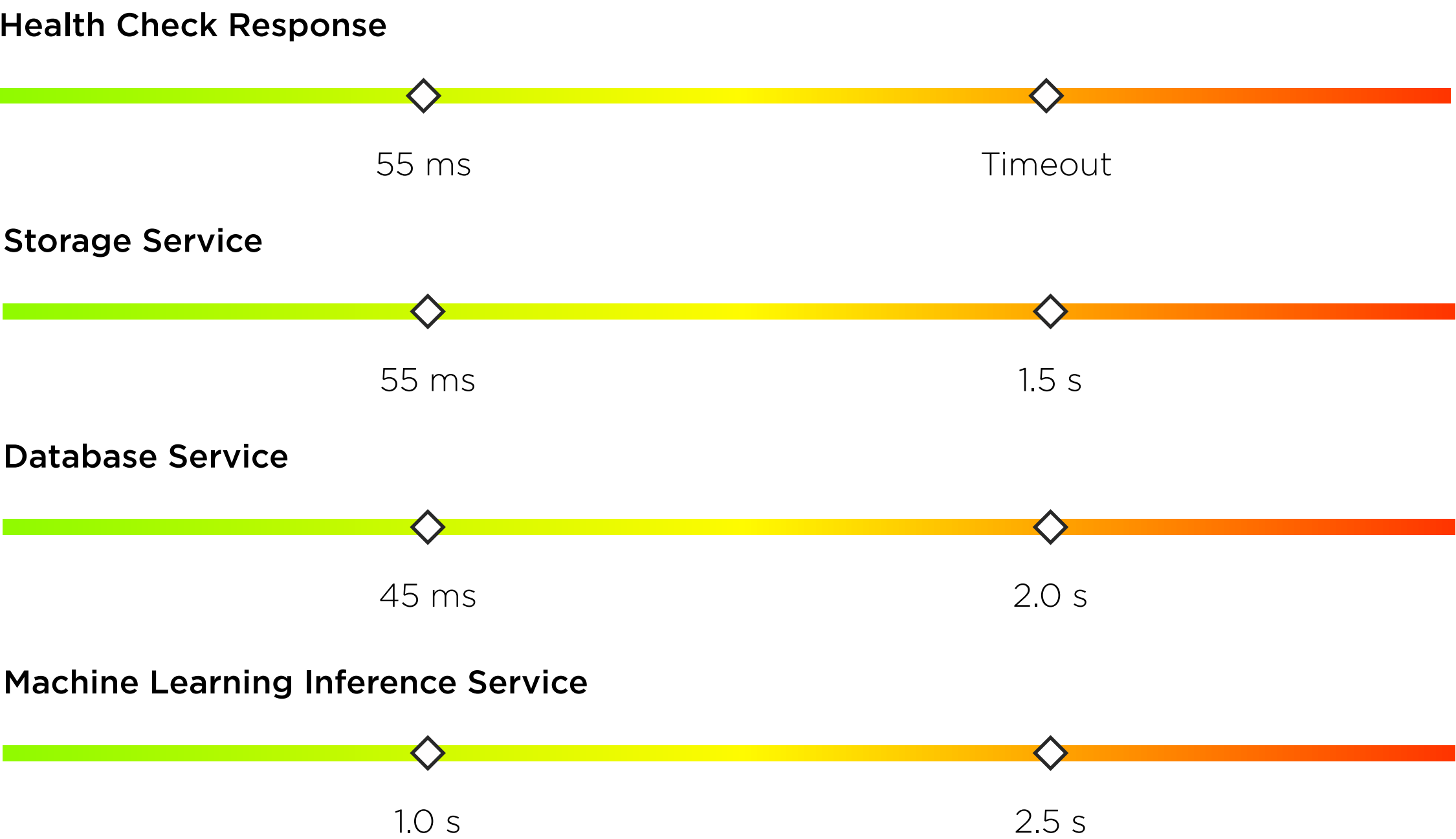
Service-level Objective Elements

Overall system availability - the percentage of time in which a service is available

Operational throughput - measure of the amount a system can handle

Operational response time - range of the time for a system to complete an operation

Application Health Points



Tracking Application Dependencies



Azure Monitor

Analyze and diagnose issues across applications with **Application Insights**

Analyze and correlate infrastructure issues with **Azure Monitor for VM's and Containers**

Utilize **Log Analytics** for a deeper level of diagnostics

Provide both **smart alerts** and **automated actions** based on system condition

Enable visualizations with **Azure dashboards** and **workbooks**

Utilize **Azure Monitor Metrics** to collect data from system resources

“... if the overall system is depicted as **partially healthy**, the operator should be able to zoom in and determine which functionality is currently unavailable.”

Microsoft Azure Documentation

Partial Health Questions

In situations where a health check indicates partial health:

- What systems are affected?
- What is the status of an affected system?
- What is the cause of the issue?

Types of Analysis

Hot Analysis

Time-critical analysis
of current state and
cause

Warm Analysis

Analysis of data
leading up to health
event

Cold Analysis

Analysis long-term
data to determine
overall trends

Dependency Tracking

Application Insights provides the ability to track and monitor calls to application-dependent systems. This can assist in determining what checks to include a health check as well as providing a tool to help with warm analysis of issues that arise.

Dependency Tracking

Application Insights SDK's for .NET and .NET Core provides automatic tracking

Supports tracking of the following dependencies:

- HTTP/HTTPS
- WCF (if HTTP bindings are used)
- SQL
- Azure Storage
- EventHub Client SDK
- ServiceBus Client SDK
- Azure Cosmos DB (if HTTP/HTTPS are used)

Manual Dependency Tracking

Some systems are not supported by default:

- Cosmos DB (if TCP is used)
- Redis

**Dependencies can be tracked manually
using the TrackDependency API**

Application Insights Dependencies

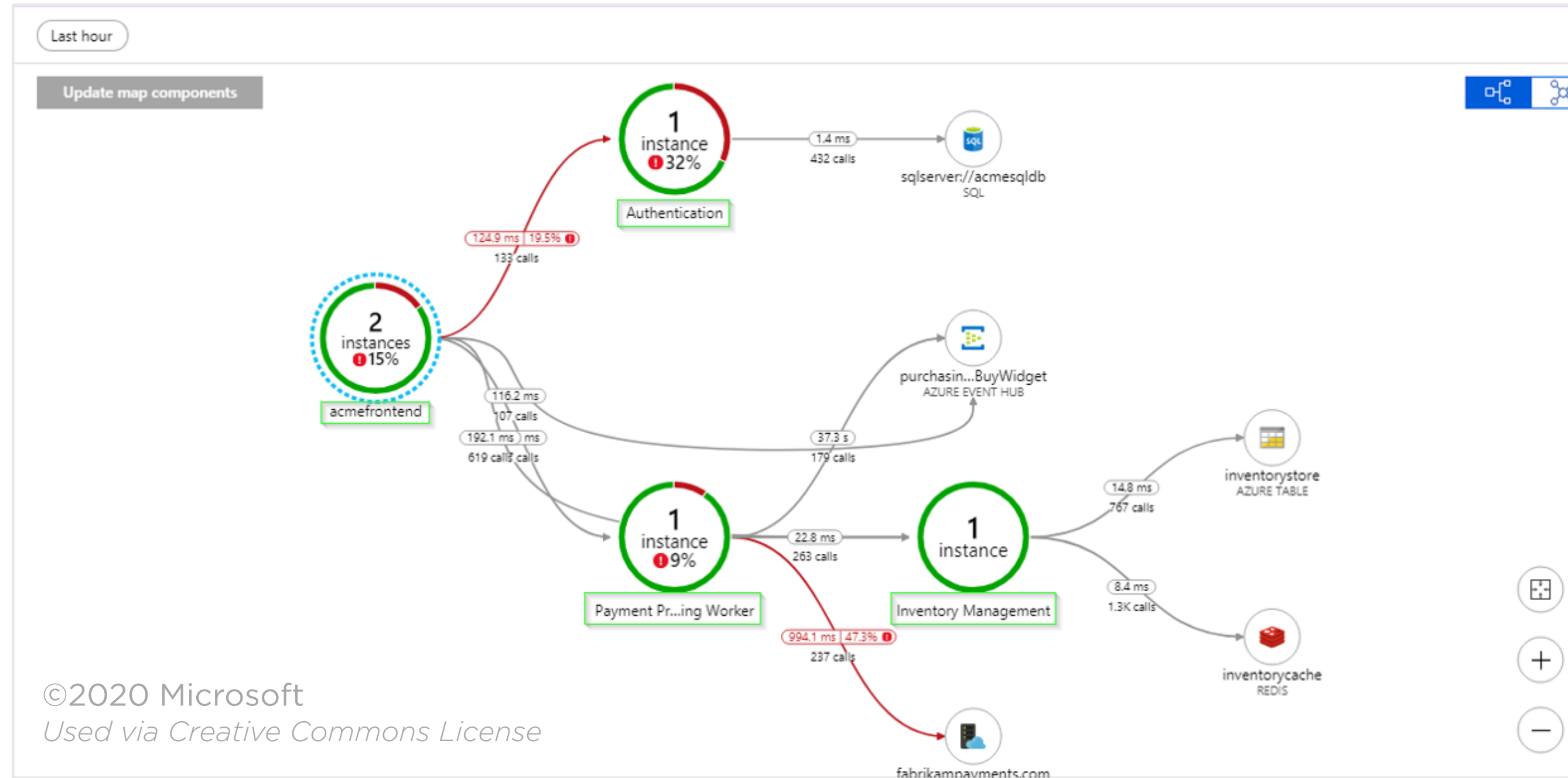
Application Map

Transaction Diagnostics

Browsers

Analytics

Application Insights Application Map



Application Insights Transaction Diagnostics

The screenshot displays the Application Insights Transaction Diagnostics interface. The main pane shows an end-to-end transaction for 'App Service Customer Details' in Central US, with Operation ID: b8e278fcfd20456a88ab48ac9cde6476. The transaction timeline includes several events: a request to 'http://fabrikamfiberapp.azurewebsites.net/Customers/Details/8469' (557 ms), a GET request to 'fabrikamfiberapp' (430.3 ms), and several database calls to 'fabrikamxyz' and 'fabrikamaccount'. An exception, 'System.FormatException', is highlighted in blue, indicating a failure in the transaction. The right-hand pane provides details about the exception, including its properties and the call stack. The call stack shows the exception was thrown in the 'ValidZipCode' method of the 'AddressValidator' class. The bottom of the interface shows a summary of 10 All, 1 Traces, and 0 Events, along with a 'View all telemetry' button.

Search results Learn more

End-to-end transaction
Operation ID: b8e278fcfd20456a88ab48ac9cde6476

Legend: Request (incoming), Dependency (outgoing), Exception, Profiler trace, Debug snapshot

EVENT	RES.	DURATION	MS
App Service Customer Details Central US		557 ms	
http://fabrikamfiberapp.azurewebsites.net/Customers/Details/8469		557 ms	
fabrikamfiberapp GET Customers/Details	500	430.3 ms	
fabrikamxyz FabrikamISQL		57 ms	
fabrikamaccount POST fabrikamaccount/Tables	409	112 ms	
fabrikamaccount POST fabrikamaccount/fabrikamfiber	409	111 ms	
fabrikamaccount GET fabrikamaccount/fabrikamfiber	200	48 ms	
EXCEPTION System.FormatException			
Failed Central US		0	

Feedback

Create work item Open debug snapshot

EXCEPTION
System.FormatException

Exception Properties [Show all](#)

Event time	8/23/2018, 6:14:49 AM
Message	Input string was not in a correct format.
Exception type	System.FormatException
Failed method	FabrikamFiber.DAL.Data.AddressValidator.ValidZipCode

Call Stack ☐ Show Just My Code

```
System.FormatException:  
at System.Number.StringToNumber (mscorlib, Version=4.0.0.0, Cult  
at System.Number.ParseInt32 (mscorlib, Version=4.0.0.0, Culture=  
at System.Int32.Parse (mscorlib, Version=4.0.0.0, Culture=neutra  
at FabrikamFiber.DAL.Data.AddressValidator.ValidZipCode (Fabrika  
at FabrikamFiber.DAL.Models.Address.FullAddress (FabrikamFiber.D  
at ASP._Page_VIEWS_Customers_Details_cshtml.Execute (App_Web_sw4  
at System.Web.WebPages.WebPageBase.ExecutePageHierarchy (System.  
at System.Web.Mvc.WebViewPage.ExecutePageHierarchy (System.Web.M  
at System.Web.WebPages.StartPage.RunPage (System.Web.WebPages, v  
at System.Web.WebPages.StartPage.ExecutePageHierarchy (System.We  
at System.Web.WebPages.WebPageBase.ExecutePageHierarchy (System.  
at System.Web.Mvc.RazorView.RenderView (System.Web.Mvc, Version=  
at System.Web.Mvc.BuildManagerCompiledView.Render (System.Web.Mv  
at System.Web.Mvc.ViewResultBase.ExecuteResult (System.Web.Mvc,  
at System.Web.Mvc.ControllerActionInvoker.InvokeActionResult /su
```

Related Items

Show what happened before and after this exception in Use...

Recovering from Failure

“Recovery time objective (RTO) is the targeted duration of time and a service level within which a business process must be restored after a disaster (or disruption) in order to avoid unacceptable consequences associated with a break in business continuity”

Wikipedia

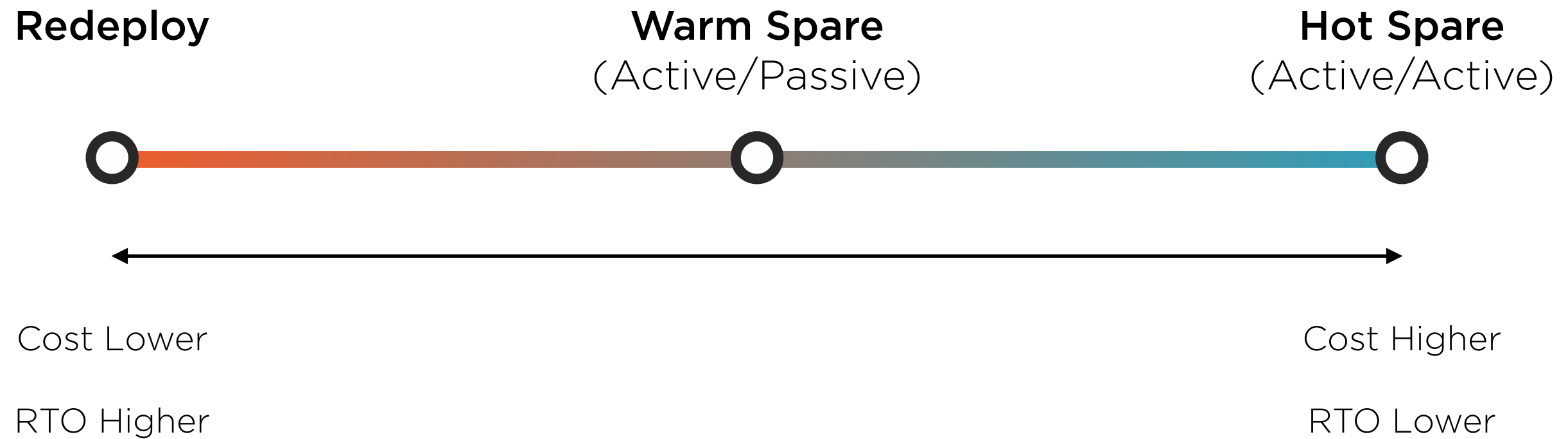
Recovery Considerations

Compute

**Application
Data**

**Managed
Services**

Application Recovery



Recovery Considerations

An effective data backup and recovery strategy is essential for a low RTO

Managed services with region replication reduce inner-region risk

Geo-replication for a service enables faster recovery when switching regions

Recovery must be practiced on a regular basis for there to be RTO confidence